Project: Bank Marketing (Campaign)

Week 10 deliverables

Group name: Fleifel-solo

Name: Rania Tarek Fleifel

Email: raniatarekfleifel@gmail.com

Country: Egypt

College: Cairo university Faculty of engineering

Specialization: Data science

Internship Batch: LISUM13: 30

Submission date: 1st December 2022

Github repo: https://github.com/RaniaFleifel/Data-glacier-internship/tree/main/Data%20science%20project_Bank%20Marketing%20Campaign/week10

Problem description

Provide ABC bank with a model that enables them to target costumers who're more probable to invest in their new term deposit product.

Picking up to where we left with the previous deliverable, we decided to hold-off scaling and removing outliers until after the EDA to make informed decisions depending on the features' relation to the target *y*.

In this section, we attempt to solidify our understanding of the features by making hypotheses and examining the data to validate or negate them. This will be handy when our model concludes to see how it compares. This also gives way to revised transformations and scaling techniques for our features. For this analysis, we keep the analysis strictly limited to the features and do not look into how the features affect the target to avoid un-intentionally manipulating the data to get better results. For better visualization, we consider the data frame after tidying features but before hot-encoding them. For a more unified structure, we represent all features in its numeric form and use Pearson correlation to guide our analysis.

EDA DISCLAIMER:

** It's crucial to note that EDA is meant to give the bank insights to further tailor their product to costumers' needs and will not be used to alter the features accordingly. Let's take the age of a costumer for example, if you're offering an 80 year old costumer the term deposit product, you don't want to offer years-long tenor to a costumer whose age is beyond the average lifetime of people in Portugal (78 in 2008). But if you have enough pointers to how costumers of this age approach their finances, you can approach him with a more agreeable plan for that phase of his life without risking investing in the wrong costumer. So this EDA attempts to relate features w.r.t each other not with the target; to avoid compromising the data fed to the predictive model, especially since the target data is severely imbalanced so insights are not conclusive for generalization.

Exploratory data analysis:

The features provided can be divided into:

a) **Personal info & personal standing of costumers**
b) **Current campaign communication with costumers**
c) **Previous contact with costumers**
d) **Date dependent economic and social indicators**

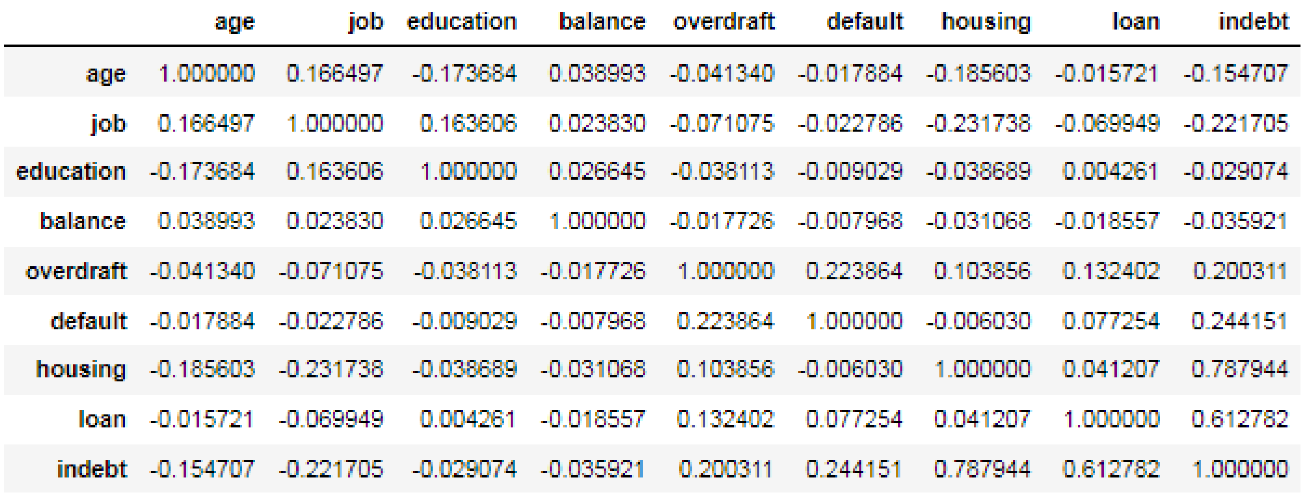
a) Personal info & personal standing:

From a bank's point of view, the ultimate goal is to target the list of costumers that are most probable to invest in this product. Hence, we attempt to relate costumers' personal data with their standing with the bank. To facilitate this analysis, we propose binning the age of costumers into *age_grp* of three groups: youth: up to 25, adult 25 to 65 and old from 65 up. These groups were chosen according to the information that the pension age in Portugal in 2008 was 65 and that it's best to obtain house mortgages starting 25 years. We also propose having a collective representative for all loans called *indebt* that simply sums the features *default,housing,loan*. These 3 features represent different indicators of loans that might affect each other but are not derivable from each other. The following table shows the dependencies inflicted on the data during feature engineering and transformation. Each of the following hypotheses deal with features from different columns.

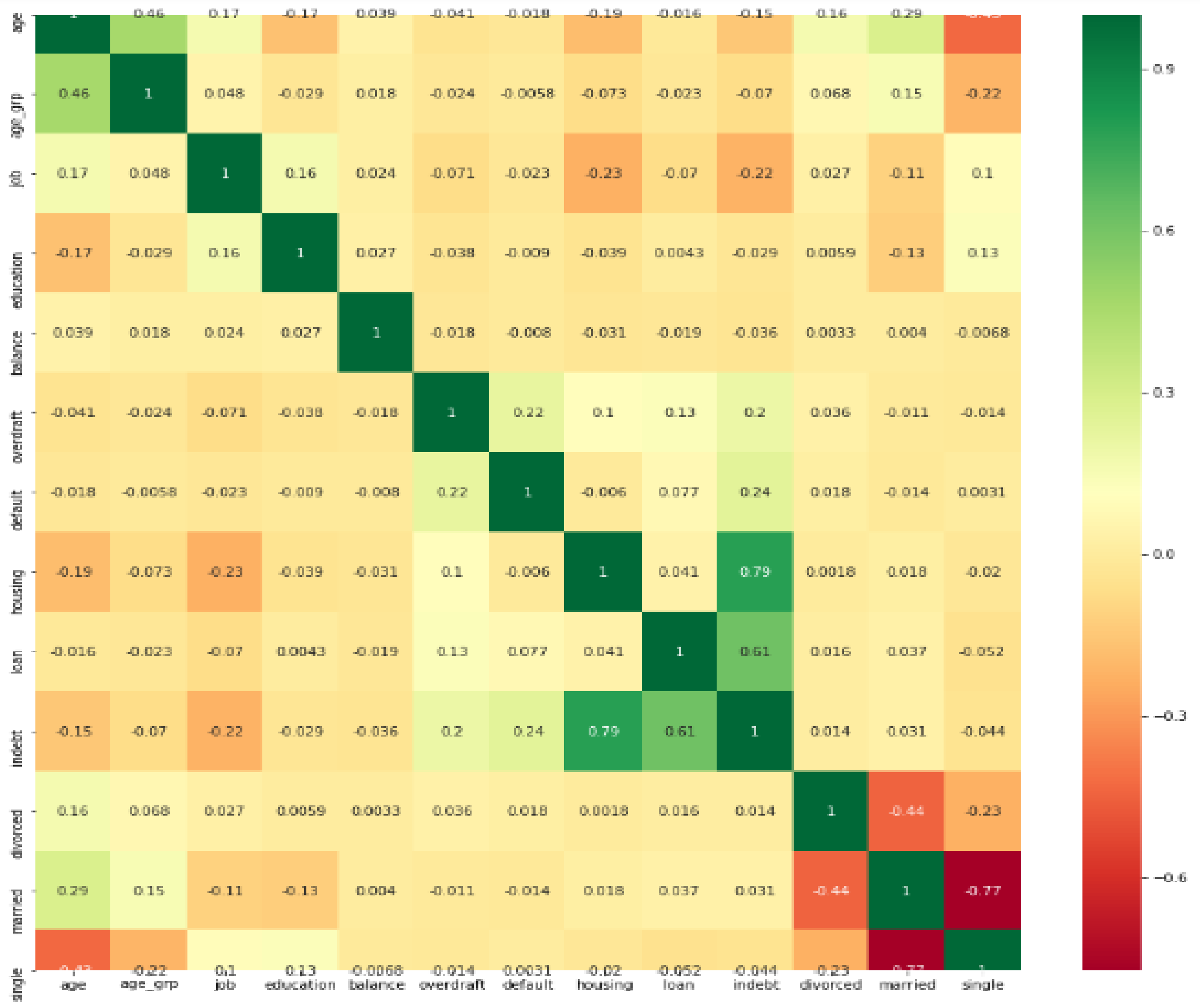
personal info → *age, job, marital, education*

personal standing related variables → *default, balance, overdraft, housing, loan*

Table of dependencies, bivariate analysis is done between variables of different columns to avoid inherent dependency						
education	*age_grp*	*job*	*indebt*	*marital*	*overdraft*	*balance*
education_missing	*age*	*job_missing*	*default, housing, loan*			

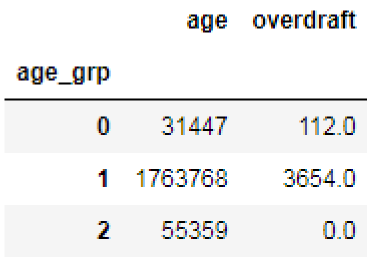


	age	job	education	balance	overdraft	default	housing	loan	indebt
age	1.000000	0.166497	-0.173684	0.038993	-0.041340	-0.017884	-0.185603	-0.015721	-0.154707
job	0.166497	1.000000	0.163606	0.023830	-0.071075	-0.022786	-0.231738	-0.069949	-0.221705
education	-0.173684	0.163606	1.000000	0.026645	-0.038113	-0.009029	-0.038689	0.004261	-0.029074
balance	0.038993	0.023830	0.026645	1.000000	-0.017726	-0.007968	-0.031068	-0.018557	-0.035921
overdraft	-0.041340	-0.071075	-0.038113	-0.017726	1.000000	0.223864	0.103856	0.132402	0.200311
default	-0.017884	-0.022786	-0.009029	-0.007968	0.223864	1.000000	-0.006030	0.077254	0.244151
housing	-0.185603	-0.231738	-0.038689	-0.031068	0.103856	-0.006030	1.000000	0.041207	0.787944
loan	-0.015721	-0.069949	0.004261	-0.018557	0.132402	0.077254	0.041207	1.000000	0.612782
indebt	-0.154707	-0.221705	-0.029074	-0.035921	0.200311	0.244151	0.787944	0.612782	1.000000



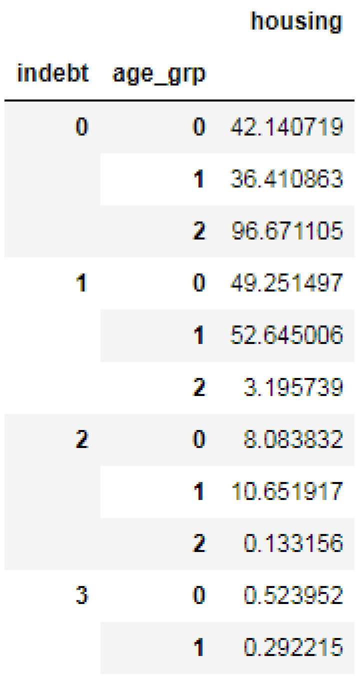
Hypotheses & Findings:

- Old people have no overdraft
  #df.groupby(['age_grp'])[['age','overdraft']].sum()



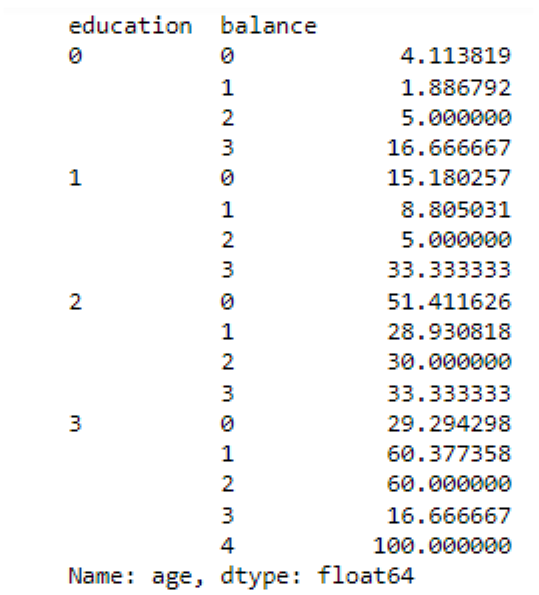
	age	overdraft
age_grp		
0	31447	112.0
1	1763768	3654.0
2	55359	0.0

- 96% of old people in the dataset have paid all their loans
  #df.groupby(['indebt','age_grp'])['housing'].count()*100/df.groupby(['age_grp'])['housing'].count()

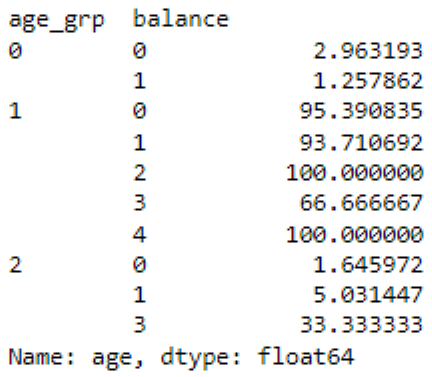


		housing
indebt	**age_grp**	
0	**0**	42.140719
	1	36.410863
	2	96.671105
1	**0**	49.251497
	1	52.645006
	2	3.195739
2	**0**	8.083832
	1	10.651917
	2	0.133156
3	**0**	0.523952
	1	0.292215

- The highest balances are associated with adults who are tertiary educated, only
  #df.groupby(['education','balance'])['age'].count()*100/df.groupby(['balance'])['age'].count()
  #df.groupby(['age_grp','balance'])['age'].count()*100/df.groupby(['balance'])['age'].count()



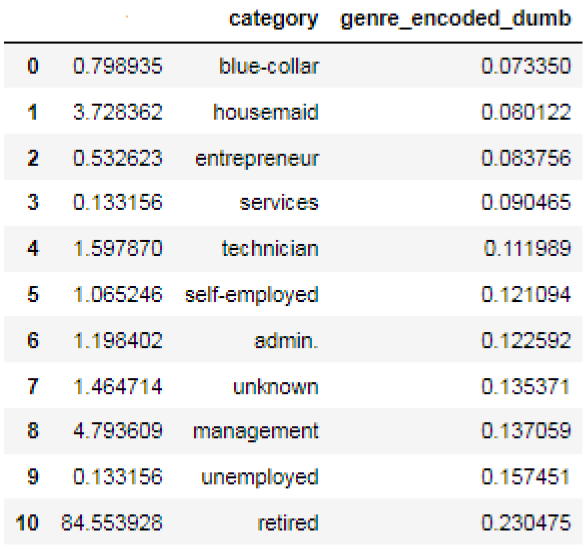
```
education  balance
0          0             4.113819
           1             1.886792
           2             5.000000
           3            16.666667
1          0            15.180257
           1             8.805031
           2             5.000000
           3            33.333333
2          0            51.411626
           1            28.930818
           2            30.000000
           3            33.333333
3          0            29.294298
           1            60.377358
           2            60.000000
           3            16.666667
           4           100.000000
Name: age, dtype: float64
```



```
age_grp  balance
0        0             2.963193
         1             1.257862
1        0            95.390835
         1            93.710692
         2           100.000000
         3            66.666667
         4           100.000000
2        0             1.645972
         1             5.031447
         3            33.333333
Name: age, dtype: float64
```

- 84% of Old people are retired, then with a landslide the most occupied jobs are management and housemaids. Services that require manual labor comes at the bottom of the list as expected

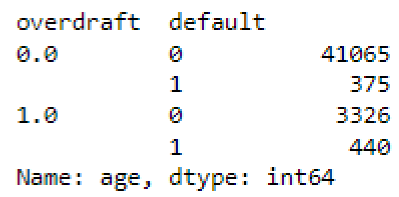
#df.query('age_grp==2').groupby(['job']).count()[['age']]*100/len(df.query('age_grp==2'))



		category	genre_encoded_dumb
0	0.798935	blue-collar	0.073350
1	3.728362	housemaid	0.080122
2	0.532623	entrepreneur	0.083756
3	0.133156	services	0.090465
4	1.597870	technician	0.111989
5	1.065246	self-employed	0.121094
6	1.198402	admin.	0.122592
7	1.464714	unknown	0.135371
8	4.793609	management	0.137059
9	0.133156	unemployed	0.157451
10	84.553928	retired	0.230475

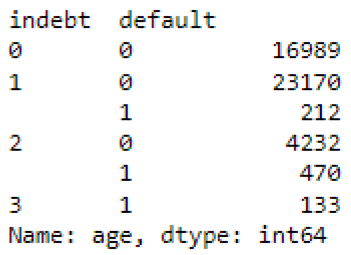
- The hypothesis was that People with no default have no overdrafts, but I was overlooking two factors. The first is that there are two types of overdraft, and while one type is related to missing payment due dates of loans, there's another one called agreed overdrafts similar to spending money from your credit card that you don't have at the moment. The second is that ,for example, someone not missing his payment's due dates doesn't mean he doesn't use credit cards at all

# df.groupby(['overdraft','default'])['age'].count()



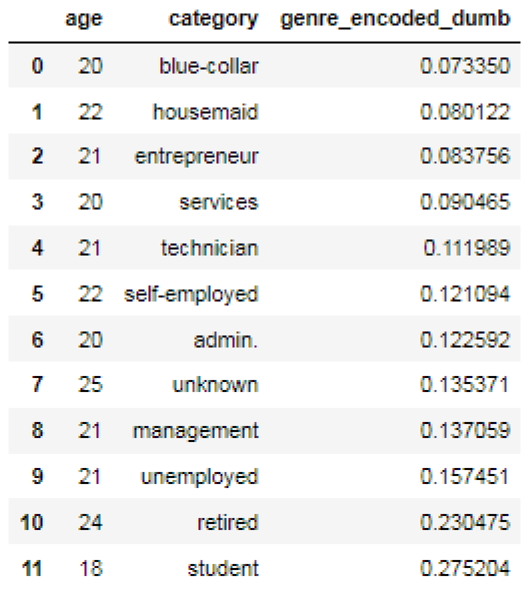
```
overdraft  default
0.0        0          41065
           1            375
1.0        0           3326
           1            440
Name: age, dtype: int64
```

- While we’re on default, people with no debts might have no default. This is based on the correlation between default and all types of loans.
  # df.groupby([‘indebt’,’default’])[‘age’].count()

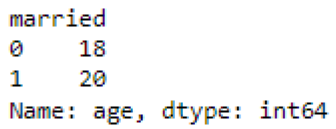


```
indebt  default
0       0          16989
1       0          23170
        1            212
2       0           4232
        1            470
3       1            133
Name: age, dtype: int64
```

- The minimum age for any job available in the dataset is 20 years, ages from 18 up to this age can only be students and not so coincidentally, 20 years is also the smallest married costumer in the dataset
  #df.groupby(['job'])['age'].min()
  #df.groupby(['married']).min().age



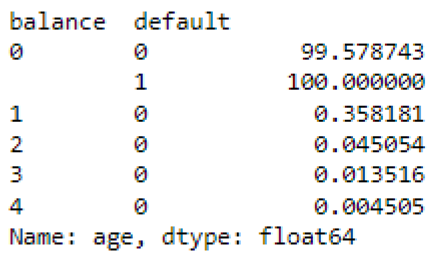
	age	category	genre_encoded_dumb
0	20	blue-collar	0.073350
1	22	housemaid	0.080122
2	21	entrepreneur	0.083756
3	20	services	0.090465
4	21	technician	0.111989
5	22	self-employed	0.121094
6	20	admin.	0.122592
7	25	unknown	0.135371
8	21	management	0.137059
9	21	unemployed	0.157451
10	24	retired	0.230475
11	18	student	0.275204



```
married
0    18
1    20
Name: age, dtype: int64
```

- Costumer’s who have default usually have small balances that doesn’t suffice paying the installments

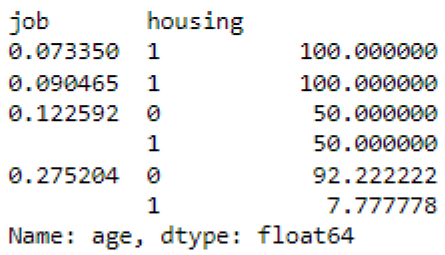
  #df.groupby(['balance','default'])['age'].count()*100/df.groupby(['default'])['age'].count()



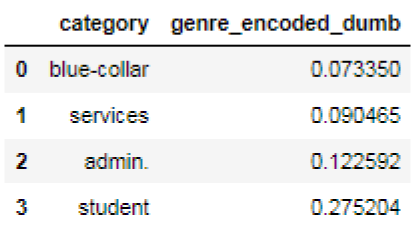
```
balance  default
0        0           99.578743
         1          100.000000
1        0            0.358181
2        0            0.045054
3        0            0.013516
4        0            0.004505
Name: age, dtype: float64
```

- Only 7% of students below 21 years have housing loans

  #df.query('age<21').groupby(['job','housing'])['age'].count()*100/df.query('age<21').groupby(['job'])['age'].count()

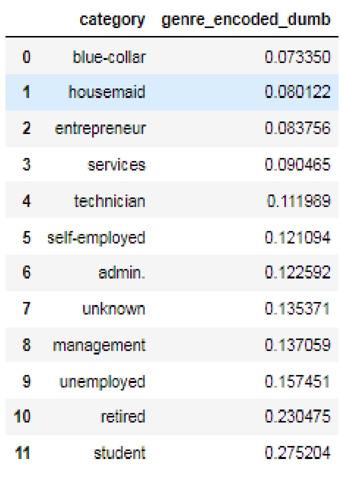


```
job       housing
0.073350  1          100.000000
0.090465  1          100.000000
0.122592  0           50.000000
          1           50.000000
0.275204  0           92.222222
          1            7.777778
Name: age, dtype: float64
```

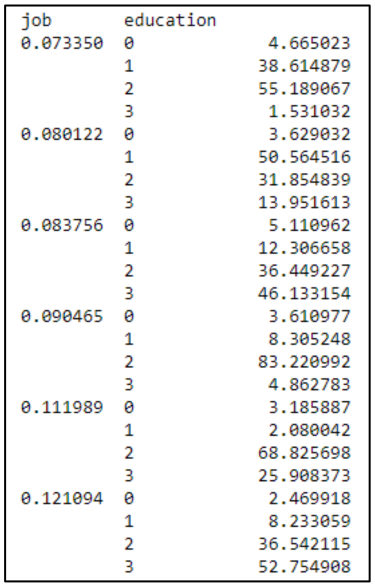


	category	genre_encoded_dumb
0	blue-collar	0.073350
1	services	0.090465
2	admin.	0.122592
3	student	0.275204

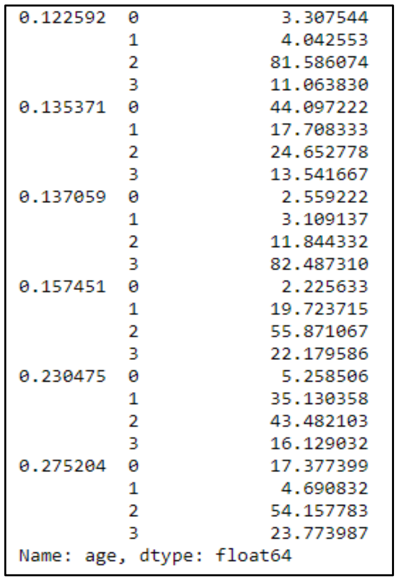
- You can tell the required education for each job using value counts or mode
  - ➔ Managerial, self-employed and entrepreneurs mostly hold tertial education
  - ➔ Admin, blue-collar, services, technician need secondary education
  - ➔ The only job that suffice with primary education is housemaid



	category	genre_encoded_dumb
0	blue-collar	0.073350
1	housemaid	0.080122
2	entrepreneur	0.083756
3	services	0.090465
4	technician	0.111989
5	self-employed	0.121094
6	admin.	0.122592
7	unknown	0.135371
8	management	0.137059
9	unemployed	0.157451
10	retired	0.230475
11	student	0.275204



```
job       education
0.073350  0            4.665023
          1           38.614879
          2           55.189067
          3            1.531032
0.080122  0            3.629032
          1           50.564516
          2           31.854839
          3           13.951613
0.083756  0            5.110962
          1           12.306658
          2           36.449227
          3           46.133154
0.090465  0            3.610977
          1            8.305248
          2           83.220992
          3            4.862783
0.111989  0            3.185887
          1            2.080042
          2           68.825698
          3           25.908373
0.121094  0            2.469918
          1            8.233059
          2           36.542115
          3           52.754908
```



```
0.122592  0            3.307544
          1            4.042553
          2           81.586074
          3           11.063830
0.135371  0           44.097222
          1           17.708333
          2           24.652778
          3           13.541667
0.137059  0            2.559222
          1            3.109137
          2           11.844332
          3           82.487310
0.157451  0            2.225633
          1           19.723715
          2           55.871067
          3           22.179586
0.230475  0            5.258506
          1           35.130358
          2           43.482103
          3           16.129032
0.275204  0           17.377399
          1            4.690832
          2           54.157783
          3           23.773987
Name: age, dtype: float64
```

- The observations made throughout this section solidify that there is an inherent cut in the age feature at ages 25 and 65.

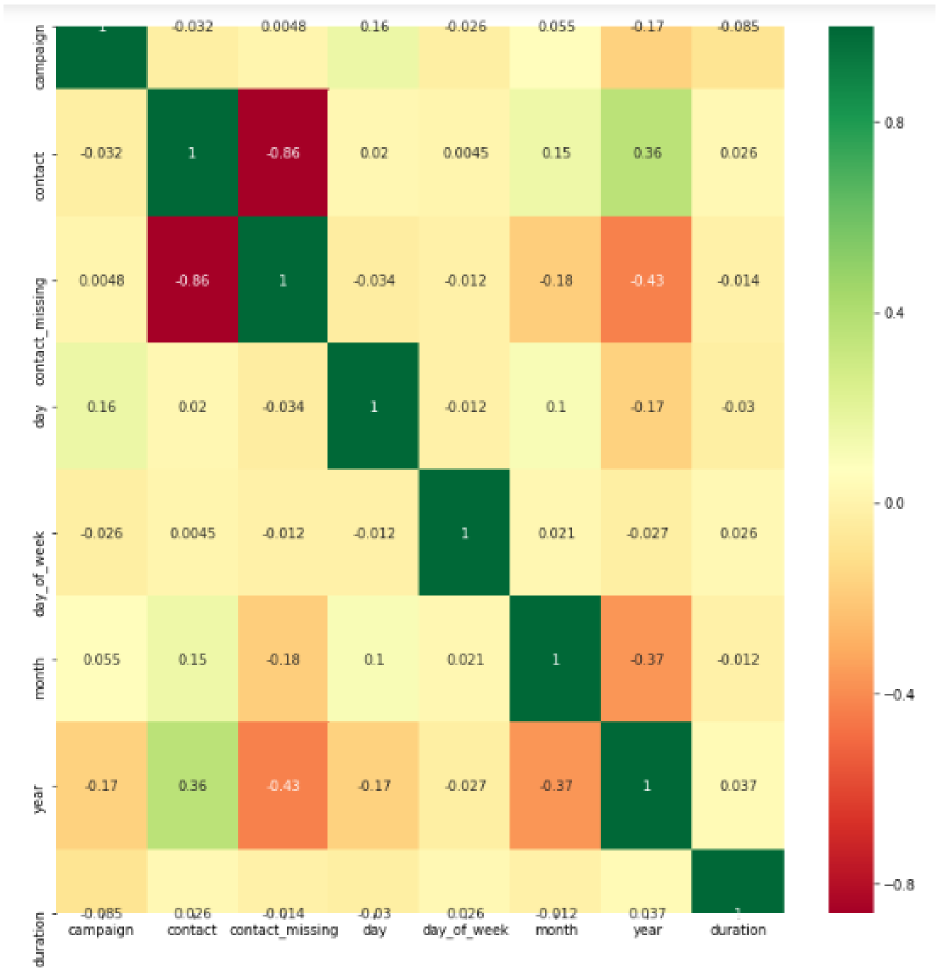
b) <u>Current campaign communication with costumers</u>

This type of data governs how the bank communicate with each costumer during the campaign at hand. The observations done on this data is expected to focus on how well costumers respond to methods of reaching out from the bank. Do people require less convincing around the time their salaries become deposited or just before that when cash is tight? Do they respond better around the holidays? Are frequent contacts with the same costumer related to less duration of contact? Is there a favorable method of contact?

this_campaign_comm → *campaign*, *contact*, *contact_missing*, *day*, *day_of_week*, *month*, *year*, *duration*.

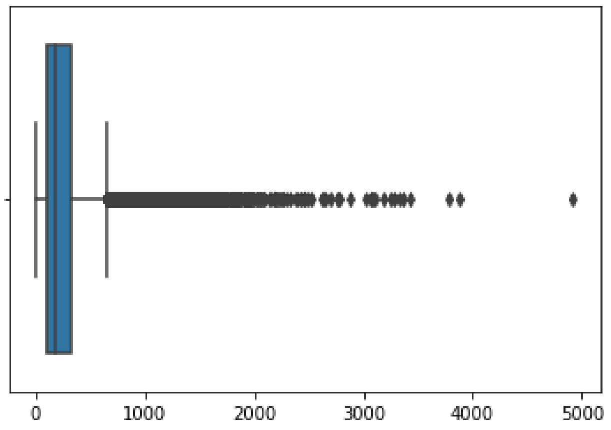
Table of dependencies, bivariate analysis is done between variables of different columns					
contact	*duration*	*day*	*campaign*	*month*	*year*
contact_missing		*day_of_week*			

	campaign	contact	contact_missing	day	day_of_week	month	year	duration
campaign	1.000000	-0.032263	0.004845	0.162528	-0.026484	0.054904	-0.166210	-0.084606
contact	-0.032263	1.000000	-0.862391	0.020170	0.004527	0.153606	0.364329	0.025518
contact_missing	0.004845	-0.862391	1.000000	-0.034091	-0.011544	-0.182433	-0.428380	-0.014356
day	0.162528	0.020170	-0.034091	1.000000	-0.011908	0.101981	-0.170305	-0.030126
day_of_week	-0.026484	0.004527	-0.011544	-0.011908	1.000000	0.020587	-0.027474	0.026442
month	0.054904	0.153606	-0.182433	0.101981	0.020587	1.000000	-0.373885	-0.011892
year	-0.166210	0.364329	-0.428380	-0.170305	-0.027474	-0.373885	1.000000	0.037028
duration	-0.084606	0.025518	-0.014356	-0.030126	0.026442	-0.011892	0.037028	1.000000



duration is the only continuous variable related to the campaign's communication with this costumer. We look into its properties to reach an appropriate way to represent it. Its high variance is not conclusive to derive insights. It's also expected that there are outliers since the difference between 75% and max is huge, we validate this with the box plot that confirms the hypothesis.

```
count    45206.000000
mean       258.153143
std        257.519164
min          0.000000
25%        103.000000
50%        180.000000
75%        319.000000
max       4918.000000
Name: duration, dtype: float64
```

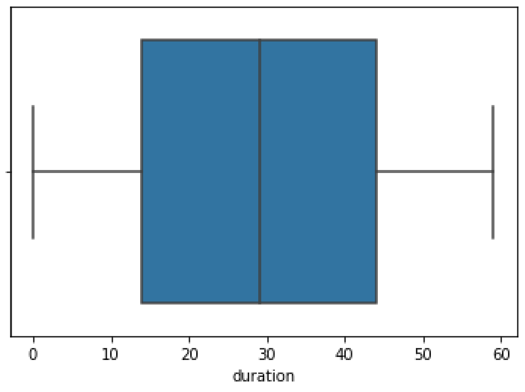


Looking into clipping or cupping the upper outliers, we look into the amount of data that we would be losing. Assuming we're looking into the 99% percentile, there's a lot of entries to lose.

```
durations statistics
upper limit=1269.0
max=4918
len(> upper_lim)=452
```

So instead of removing outliers, we attempt to represent the variable in a more generic manner. This is straight forward through defining the call duration through minutes instead of seconds. This hides a lot of unnecessary details and maintains all data intact.

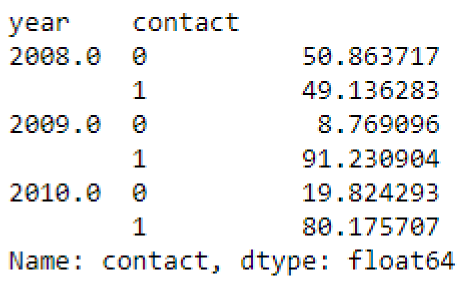
```
<matplotlib.axes._subplots.AxesSubplot at 0xbc691c8
```



It's crucial to look at duration from a contextual point of view as well. The duration of a call with a costumer is not data that would bs used beforehand. But rather after a call, when it's already obvious whether the costumer went for the deposit product or not.

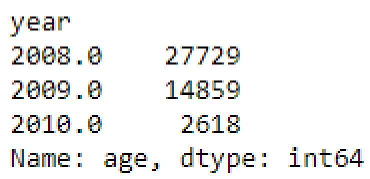
We look at the following statistics year by year to discuss overall governing relation without overanalyzing details that might be coincidental over a smaller time interval.

- Method of contact per year
  The relatively high correlation between contact and year gauges us to look into it. It makes sense that the contact method moves toward cellular as years pass by.



```
year    contact
2008.0  0          50.863717
        1          49.136283
2009.0  0           8.769096
        1          91.230904
2010.0  0          19.824293
        1          80.175707
Name: contact, dtype: float64
```

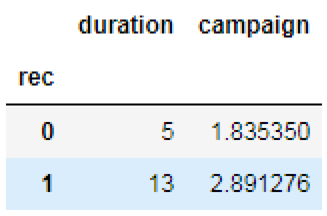
- Costumers contacted per year



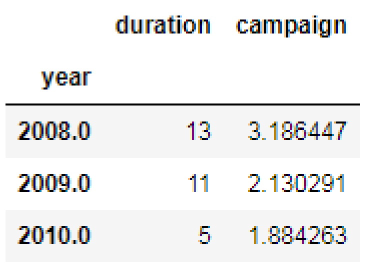
```
year
2008.0    27729
2009.0    14859
2010.0     2618
Name: age, dtype: int64
```

It's interesting that the data starts in 2008 in May, and yet the number of costumers contacted makes up almost 60 % of the data. Let's note that 2008 is right in the middle of the recession period that lasted till June 2009. So we add a variable *rec* that is a binary representation of 1 untill June 2009 then is 0

- Mode of *duration* and *mean* campaign per *year* and per *rec* variables



	duration	campaign
rec		
0	5	1.835350
1	13	2.891276

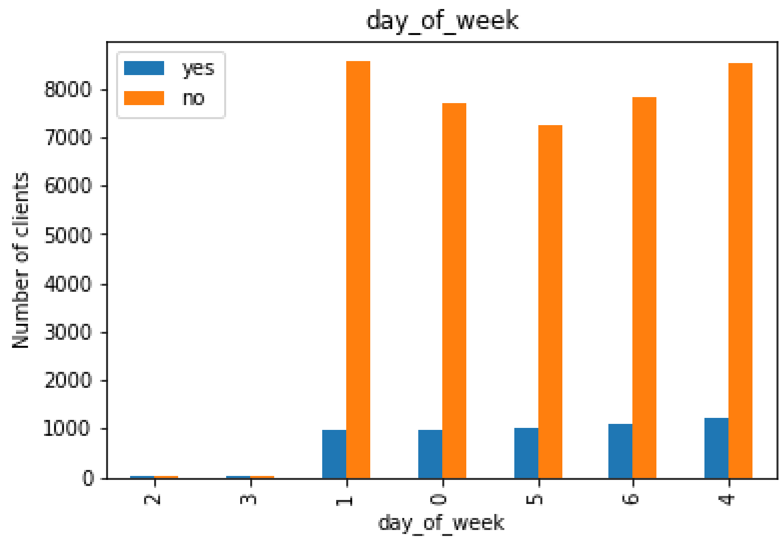


	duration	campaign
year		
2008.0	13	3.186447
2009.0	11	2.130291
2010.0	5	1.884263

It makes sense that the frequency of contacting costumers, whether we're talking about number of calls or the durations, is highest at the peak of the recession because it was an absolutely necessary product given the bad economy .

- Mode of *day_of_week*

  It looks like bank employees focus their communication with costumers away from the middle of the week.

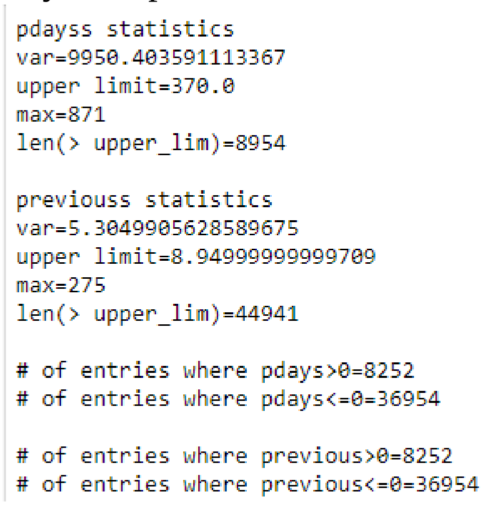


c) <u>Previous contact with costumers</u>

previous_contact → pdays, previous, poutcome, poutcome_missing

The data in this group is particularly challenging to read into because

- There's strong dependency between all variables since we added 'non_existant' value to poutcome based on pdays and previous.
- Both pdays and previous are continuous variables.

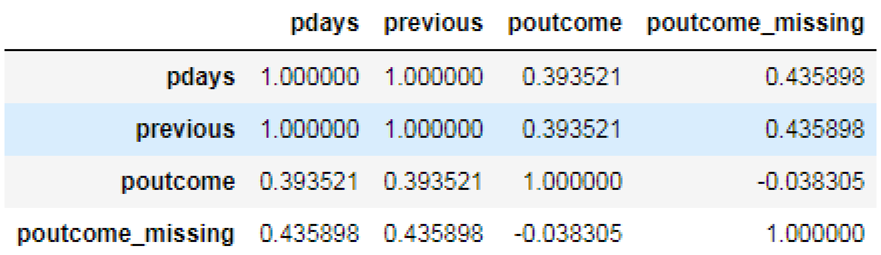


```
pdayss statistics
var=9950.403591113367
upper limit=370.0
max=871
len(> upper_lim)=8954

previouss statistics
var=5.3049905628589675
upper limit=8.94999999999709
max=275
len(> upper_lim)=44941

# of entries where pdays>0=8252
# of entries where pdays<=0=36954

# of entries where previous>0=8252
# of entries where previous<=0=36954
```

The statistics show lots of outliers in *previous* and huge variance in *pdays*! As seen from the statistics, truncating outliers will cause a huge loss in information and binning it will be arbitrary as well. At this point, we're inclined to changing both variables to a binary indicator that's 0 when the original data is 0 and 1 otherwise. Looking into the number of entries we're giving up numeric details in, we find that the number of those is still significantly lower than what we'd lost if we truncate outliers.

# pdays to a binary 0 if pdays=0 , 1 otherwise

# previous to a binary 0 if pdays=0 , 1 otherwise

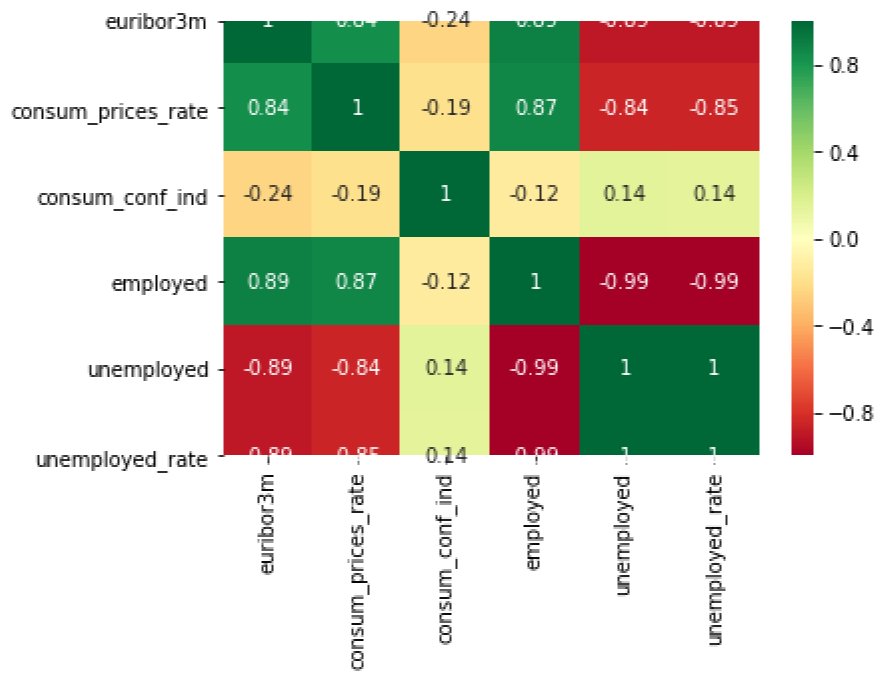


	pdays	previous	poutcome	poutcome_missing
pdays	1.000000	1.000000	0.393521	0.435898
previous	1.000000	1.000000	0.393521	0.435898
poutcome	0.393521	0.393521	1.000000	-0.038305
poutcome_missing	0.435898	0.435898	-0.038305	1.000000

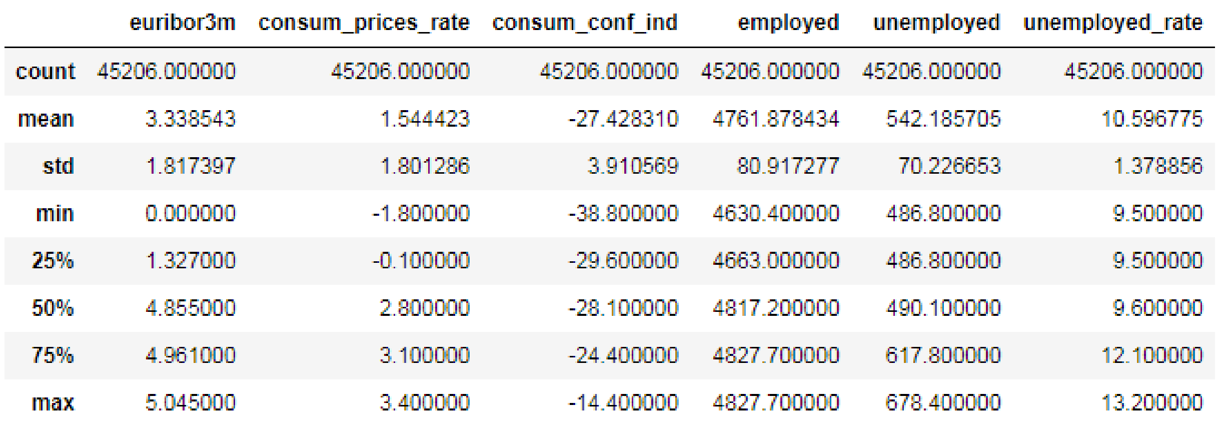
After using binary values to represent the two variables, they act identically and hence we can now safely remove one of them, for instance *pdays*

d) <u>Date dependent economic and social indicators</u>

eco_indic→ *euribor3m*, *consum_prices_rate*, *consum_conf_ind*, *employed*, *unemployed*, *unemployed_rate*



From the correlation matrix alone, it's obvious that the 3 variables *employed,unemployed,unemployed_rate* are dependent. We choose to keep only unemployed_rate among the three for its reasonable mean and var, especially that its underlying include the two removed variables.



	euribor3m	consum_prices_rate	consum_conf_ind	employed	unemployed	unemployed_rate
count	45206.000000	45206.000000	45206.000000	45206.000000	45206.000000	45206.000000
mean	3.338543	1.544423	-27.428310	4761.878434	542.185705	10.596775
std	1.817397	1.801286	3.910569	80.917277	70.226653	1.378856
min	0.000000	-1.800000	-38.800000	4630.400000	486.800000	9.500000
25%	1.327000	-0.100000	-29.600000	4663.000000	486.800000	9.500000
50%	4.855000	2.800000	-28.100000	4817.200000	490.100000	9.600000
75%	4.961000	3.100000	-24.400000	4827.700000	617.800000	12.100000
max	5.045000	3.400000	-14.400000	4827.700000	678.400000	13.200000

Summing up the adjustments to the variables made during EDA

- Add '*indebt*'
- Bin '*age*' into '*age_grp*'
- Add '*rec*'
- Binary represent '*previous*'
- Drop '*pdays*'

- Using minutes representation for ‘*duration’*
- Drop *employed* and *unemployed* variables

Adding these adjustments to the ones done during data manipulation and feature engineering provide the features to feed to the predictive model.